PATENT
Attorney Docket No. GB920000068US1

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of

Michele CRUDELE et al.

Serial No: 09/766,811

Filed: January 22, 2001

For: PREFERABLE MODES OF SOFTWARE PACKAGE DEPLOYMENT

Examiner: Tuan A. VU

Art Unit: 2124

CERTIFICATE OF SUBMISSION BY FACSIMILE

PTO FAX NUMBER: 703-872-9306
TOTAL NUMBER OF PAGES: 16

Dear Sir:

I hereby certify that the following documents are being transmitted to the U.S. Patent and Trademark Office on the date shown below:

1. APPEAL BRIEF (15 pages); and

2. This CERTIFICATE OF SUBMISSION BY FACSIMILE (1 page).

If you did not receive all the pages, please telephone us at 718-544-1110, or fax us at 718-544-8588.

Respectfully submitted,

Dated: March 24, 2005

Ido Tuchman, Reg. No. 45,924
69-60 108th Street, Suite 503
Forest Hills, NY 11375
Telephone (718) 544-1110
Facsimile (718) 544-8588

PATENT
Attorney Docket No. GB920000068US1

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of

Michele CRUDELE et al.

Serial No: 09/766,811

Filed: January 22, 2001

For: PREFERABLE MODES OF SOFTWARE PACKAGE DEPLOYMENT

Examiner: Tuan A. VU

Art Unit: 2124

APPEAL BRIEF

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

The Appellant submits this brief pursuant to 37 C.F.R. §1.192 in furtherance of the Notice of Appeal timely filed in this case on January 24, 2005, setting a two-month shortened statutory period of brief filing expiring March 24, 2005.

Please charge Deposit Account 50-0510 the $500 fee for filing this Appeal Brief. No other fee is believed due with this Appeal Brief, however, should another fee be required please charge Deposit Account 50-0510.

Real Party In Interest

The real party in interest is International Business Machines Corporation.

Related Appeals And Interferences

None.

Status of Claims

Claims 1-12 are pending in the present application (see Appendix A for a listing of the claims).

Claims 1-5 stand rejected under 35 U.S.C. §103(a) as unpatentable over U.S. Patent No. 5,845,090 to Collins, III et al. (hereinafter "Collins") in view of U.S. Patent No. 6,505,228 to Schoening et al. (hereinafter "Schoening"), SmartUpdate, "SmartUpdate Developer's Guide", http://developer.netscape.com/docs/manuals/communicator/jarman/install.htm (hereinafter "SmartUpdate"), and U.S. Patent No. 6,389,589 to Mishra et al. (hereinafter "Mishra").

Claims 6-12 stand rejected under 35 U.S.C. §103(a) as unpatentable over Collins in view of Mishra.

Status of Amendments

Additional claims 13-17 introduced by the Amendment and Response to Final Office Action filed October 18, 2004 were not entered by the Examiner.

Summary of the Invention

The present invention provides a means of managing and distributing software. Application, page 1, lines 4-5. Software deployment can be a challenge in large distributed networks containing thousands of systems with multiple platforms. Application, Fig. 17. The present invention provides an efficient and reliable solution to deploy software packages. Application, page 3, lines 14-16.

In accordance with an embodiment of the invention, a software package to be deployed comprises a hierarchical structure of leaf and branch nodes capable of being traversed in a top-down manner. Application, page 7, lines 6-8 and Fig. 2. Each leaf node of the software package hierarchy corresponds to an action (i.e., restart, add file, remove file) to be performed on a target

system. Application, page 7, lines 9-11. It is contemplated that the software package can be represented by a sequence of stanzas, with each stanza representing an action. Application, page 17, lines 21-22.

Once the software package is distributed, it is available to a target endpoint. An engine resident on the target can be instructed via a management agent to decode the hierarchical software package from the file into memory and then to perform various software distribution operations including installing, removing and modifying the software and configuration of the endpoint. Application, page 6, line 22 - page 7, line 4.

As the hierarchical software package file is read, each node causes a corresponding class to be instantiated. Application, page 7, lines 22-26. More specifically, for each type of action in the software package, a class is instantiated and its attributes are set to correspond to the action defined in the software package file. Application, page 8, lines 23-25.

There are typically several methods to deploy a software package, such as Basic, Transactional, Undoable, and Undoable in Transactional. Application, page 33, lines 3-6. In some cases, an administrator may prefer one deployment method to a second, less preferable method. Application, page 43, lines 3-5. In one embodiment of the invention, an engine resident at the target checks whether the preferable deployment method is possible and performs the preferable deployment method upon a successful check. Application, page 43, lines 14-19. If the check is unsuccessful, the engine performs the less preferable deployment method. Application, page 43, lines 9-14.

Issues

1. Whether claims 1-5 are unpatentable under 35 U.S.C. §103(a) over Collins in view of Schoening, SmartUpdate, and Mishra.

2. Whether claims 1-2 are unpatentable under 35 U.S.C. §103(a) over Collins in view of Mishra.

Grouping of Claims

All claims are grouped together for the purpose of this appeal.

Argument

I. Collins, Schoening, SmartUpdate and Mishra do not teach all the limitations of claims 1-5

Claim 1

In rejecting claims under U.S.C. §103, the Examiner bears the initial burden of establishing a *prima facie* case. In re Oetiker, 977 F.2d 1443, 1445, 24 USPQ 1443, 1444 (Fed. Cir. 1992). To establish *prima facie* obviousness of a claimed invention, all the claim limitations must be taught or suggested by the prior art. In re Royka, 490 F.2d 981, 180 USPQ 580 (CCPA 1974). "All words in a claim must be considered in judging the patentability of that claim against the prior art." In re Wilson, 424 F.2d 1382, 1385, 165 USPQ 494, 496 (CCPA 1970).

Claim 1 of the present Application stands rejected as obvious over Collins in view of Schoening, SmartUpdate and Mishra. Final Office Action, page 2.

Claim 1 recites, in part, "means for executing a check method on at least one of each of said at least one class instances to determine if a deployment operation can be implemented in a specified first mode." Claim 1, lines 17-19. The Examiner apparently contends that this claim element is disclosed in Fig. 8 of Collins, stating, "means for executing a check method on one of said object actions/methods to determine if a deployment can be implemented in a first mode (e.g. Fig. 8)." Final Office Action, page 3.

The Appellant respectfully disagrees with the Examiner's interpretation of Fig. 8 in Collins. A cursory inspection of Fig. 8 reveals that the figure is devoid of any teaching or suggestion that the state machine depicted is responsive to a check method to determine if a deployment can be implemented in a first mode. Moreover, the description of Fig. 8 indicates that this figure shows the status of a generic software package changing from unbuilt to clean to ready to closed status at a central package manager. Collins, col. 8, lines 39-67. Clearly Fig. 8 and its accompanying description have nothing to do with checking if a deployment operation can be implemented.

The Examiner seems to assume that what Collins describes in Fig. 8 deals with what happens to the package when deployed on remote agents. This appears to be incorrect since what is described is the status of a generic software package residing at the central package manager. Collins, col. 8, lines 39-42. This has nothing to do with deployment actions on targets (installation, undo ability, etc.), but merely server-side preparation steps.

Claim 1 also recites, in part, "means, responsive to check failure of any class instance, for executing a second method on each of said at least one class instances in a second mode, the second mode being less preferable than the first mode." Claim 1, lines 24-27. The Examiner apparently argues that this claim element is disclosed in Fig. 8 columns 8 and 9 of Collins, stating, "means responsive to check failure of any object actions, for executing said actions in a second less preferable mode that the first method (e.g. UNBUILT - Fig. 8; restore-col. 8, line 39 to col.9, line 7-Note: a restore instance is a uninstall mode)." Final Office Action, page 3.

The Appellant again respectfully disagrees with the Examiner's interpretation of Collins. Nowhere in Collins is there a teaching or suggestion of a second mode that is less preferable than a first mode. As pointed out above, the Collins citation

offered by the Examiner merely describes the states that a package can go through as it is being prepared on a central package manager. The Collins citation does not discuss checking for failures of class instances or executing a second method on each class instances in a second, less preferable, second mode.

The Examiner, although acknowledging that Collins does not disclose a second mode being less preferable than a first mode, states, "this association of mode of install with some install method as well as uninstall method with a less desirable uninstall method has been implicitly disclosed in the teaching of Collins as well as obvious in light of the combined Collins, SmartUpdate and Mishra's teachings." Final Office Action, page 6. For the reasons given above, the Appellant respectfully disagrees with the Examiner's finding that any of the references, either along or in combination, teach a second mode being less preferable than a first mode, as claimed in claim 1. Furthermore, the Appellant points out that no evidence has been offered by the Examiner beyond conclusory statements in reaching such a finding.

Claim 1 further recites, "the software package comprises a hierarchical structure of leaf and branch nodes capable of being traversed in a top-down manner and each leaf node corresponding to said respective type of action." Claim 1, lines 8-11. The Examiner states that Collins does not explicitly disclose a package comprising a hierarchical structure of leaf and branch nodes, but points to Fig. 4 and col. 5, lines 1-10 of Collins as teaching actions "similar to parsing an algorithm". Final Office Action, pages 4-5.

In response, that Appellant respectfully submits that Fig. 4 and column 5, lines 1-10 of Collins make no mention of hierarchical structures of any kind. Fig. 4 of Collins shows the various actions taken depending on the type of packages received. Collins, col. 6, lines 51-54. Furthermore, column 5, lines 1-10 of Collins is completely unrelated to the topic of hierarchical structures, reading,

> Files, scripts and data are drawn from this file system by using a software program called a Package Tool (8). The Package Tool enables the user to specify the following attributes for each package: (1) name, description and identifier for the package, (2) type of package (distribution, collection, or command), (3) name of author, (4) specific attributes of the methods, and (5) an optional assigned profile.
>
> The Package Tool also enables the user to specify the package contents, including data files and methods. Collins, col. 5, lines 1-10.

The Examiner also alludes that the Jar file and scripts taught by SmartUpdate suggest "a certain level of hierarchy among the components being packaged." Final Office Action, page 5. The Appellant submits that SmartUpdate does not mention or suggest a software package including a hierarchical structure of leaf and branch nodes capable of being traversed from parent to child in a top-down manner, as claimed in claim 1. SmartUpdate is focuses on using and extending a sample javascript logic that can be executed on each Netscape Communicator web browser, with some java-specific installation logic. This allows either driving installation achieved by an external binary executable, or directly extracting deployment artifacts (browser plug-ins or signed java programs) from the jar being downloaded into Communicator's folders.

The Examiner also mentions that Mishra teaches software packages comprising a hierarchical structure of leaf and branch nodes capable of being traversed in a top-down manner and each leaf node corresponding to the respective type of action. Final Office Action, page 5. The Appellant respectfully disagrees with these conclusions as well.

Contrary to the Examiner's assertions, Mishra does not teach or suggest a software package including a hierarchical structure of leaf and branch nodes capable of being traversed from parent to child in a top-down manner, as claimed in claim 1. Mishra makes mention of hierarchal organization, but this teaching refers to a hierarchy of domains and organizational units in which objects are placed, not software packages themselves. Mishra, col. 4, lines

41-54 and lines 53-57. Therefore, the Appellant respectfully submits the hierarchical structure referred to by Mishra is completely different to the hierarchical structure of a software package recited in claim 1 of the present Application.

In the Office Action, the Examiner cites the following passages of Mishra as apparently teaching the claim limitations cited above: Table 3, col. 7; Figs. 3 and 5B; and the entire text from col. 6, line 25 to col. 11, line 30. Office Action, paragraph 4, page 5. The Appellant respectfully disagrees with such a conclusion.

Table 3 at col. 7 describes a package container object and contains no teaching or suggestion of hierarchical structure of leaf and branch nodes. Mishra, col. 6, lines 66-67 and Table 2, col. 7.

Likewise, Fig. 3 of Mishra does not show a software package having at least one file and including a hierarchical structure of leaf and branch nodes. Fig. 3 illustrates various containers under a "class store," and a class store, unlike a software package stores the state of deployment of managed application, not package files. Mishra, col. 5, lines 9-11 and col. 6, line 14. Fig. 5B and the entire text from col. 6, line 25 to col. 11, line 30 appear entirely off topic and make no mention or suggestion of a software package having at least one file and including a hierarchical structure of leaf and branch nodes.

The Appellant respectfully submits that a *prima facie* obviousness rejection, where the prior art references teach or suggest all the claim limitations, has not been established for claim 1. *See* MPEP §2143. Therefore, the Appellant submits that the rejection of claim 1 is improper and respectfully requests that the rejection of claim 1 be reversed by the honorable Board.

Claims 2-5

If an independent claim is non-obvious under 35 U.S.C. 103, then any claim depending therefrom is non-obvious. In re Fine, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988).

Claims 2-5 are dependent on and further limit claim 1. Since claim 1 is believed allowable for the reasons discussed above, claims 2-5 are also believed allowable for at least the same reasons as claim 1. Therefore, the Appellant respectfully requests that the rejections of claims 2-5 be reversed by the honorable Board.

II. Collins and Mishra do not teach all the limitations of claims 6-12

Claim 6:

Claim 6 was rejected under 35 USC §103 as obvious over Collins in view of Mishra. Final Office Action, page 7.

A *prima facie* case for obviousness can only be made if the combined prior art reference documents teach or suggest all the claim limitations. MPEP 2143.

Claim 6 recites a system for deploying software comprising, in part, a software package having at least one file and including "a hierarchical structure of leaf and branch nodes capable of being traversed from parent to child in a top-down manner." Application, claim 6.

The Examiner states that Collins does not explicitly disclose that a package comprising a hierarchical structure of leaf and branch nodes, but points to Fig. 4 and col. 5, lines 1-10 of as teaching actions "similar to parsing an algorithm". Final Office Action, pages 4-5 and page 7.

In response, that Appellant respectfully submits that Fig. 4 and column 5, lines 1-10 of Collins make no mention of hierarchical structures of any kind. Fig. 4 of Collins shows the various actions taken depending on the type of packages received.

Collins, col. 6, lines 51-54. Furthermore, column 5, lines 1-10 of Collins is completely unrelated to the topic of hierarchical structures, reading,

> Files, scripts and data are drawn from this file system by using a software program called a Package Tool (8). The Package Tool enables the user to specify the following attributes for each package: (1) name, description and identifier for the package, (2) type of package (distribution, collection, or command), (3) name of author, (4) specific attributes of the methods, and (5) an optional assigned profile.
>
> The Package Tool also enables the user to specify the package contents, including data files and methods. Collins, col. 5, lines 1-10.

The Examiner also alludes that the Jar file and scripts taught by SmartUpdate suggest "a certain level of hierarchy among the components being packaged." Final Office Action, page 5. The Appellant submits that SmartUpdate does not mention or suggest a software package including a hierarchical structure of leaf and branch nodes capable of being traversed from parent to child in a top-down manner, as claimed in claim 6. SmartUpdate is focuses on using and extending a sample javascript logic that can be executed on each Netscape Communicator web browser, with some java-specific installation logic. This allows either driving installation achieved by an external binary executable, or directly extracting deployment artifacts (browser plug-ins or signed java programs) from the jar being downloaded into Communicator's folders.

The Examiner also mentions that Mishra teaches software packages comprising a hierarchical structure of leaf and branch nodes capable of being traversed in a top-down manner and each leaf node corresponding to the respective type of action. Final Office Action, page 5. The Appellant respectfully disagrees with these conclusions as well.

Contrary to the Examiner's assertions, Mishra does not teach or suggest a software package including a hierarchical structure of leaf and branch nodes capable of being traversed from parent to child in a top-down manner, as claimed in claim 6. Mishra makes

mention of hierarchal organization, but this teaching refers to a hierarchy of domains and organizational units in which objects are placed, not software packages themselves. Mishra, col. 4, lines 41-54 and lines 53-57. Therefore, the Appellant respectfully submits the hierarchical structure referred to by Mishra is completely different to the hierarchical structure of a software package recited in claim 1 of the present Application.

In the Office Action, the Examiner cites the following passages of Mishra as apparently teaching the claim limitations cited above: Table 3, col. 7; Figs. 3 and 5B; and the entire text from col. 6, line 25 to col. 11, line 30. Office Action, paragraph 4, page 5. The Appellant respectfully disagrees with such a conclusion.

Table 3 at col. 7 describes a package container object and contains no teaching or suggestion of hierarchical structure of leaf and branch nodes. Mishra, col. 6, lines 66-67 and Table 2, col. 7.

Likewise, Fig. 3 of Mishra does not show a software package having at least one file and including a hierarchical structure of leaf and branch nodes. Fig. 3 illustrates various containers under a "class store," and a class store, unlike a software package stores the state of deployment of managed application, not package files. Mishra, col. 5, lines 9-11 and col. 6, line 14. Fig. 5B and the entire text from col. 6, line 25 to col. 11, line 30 appear entirely off topic and make no mention or suggestion of a software package having at least one file and including a hierarchical structure of leaf and branch nodes.

The Appellant respectfully submits that a *prima facie* obviousness rejection, where the prior art references teach or suggest all the claim limitations, has not been established for claim 1. *See* MPEP §2143. Therefore, the Appellant submits that the rejection of claim 1 is improper and respectfully requests that the rejection of claim 1 be reversed by the honorable Board.

Claims 7-12

If an independent claim is non-obvious under 35 U.S.C. 103, then any claim depending therefrom is non-obvious. In re Fine, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988).

Claims 7-12 are dependent on and further limit claim 6. Since claim 6 is believed allowable for the reasons discussed above, claims 7-12 are also believed allowable for at least the same reasons as claim 6. Therefore, the Appellant respectfully requests that the rejections of claims 7-12 be reversed by the honorable Board.

Conclusion

In view of the foregoing, Appellant submits that the rejections of claims 1-12 are improper and respectfully requests that the rejections of claims 1-12 be reversed by the honorable Board.

Respectfully submitted,

Dated: March 24, 2005

Ido Tuchman, Reg. No. 45,924
69-60 108th Street, Ste. 503
Forest Hills, NY 11375
Telephone (718) 544-1110
Facsimile (718) 544-8588

Appendix A

Pending Claims

Claim 1. A software deployment tool stored on a computer readable storage medium for use with a software package including a software package file incorporating at least one action defining respective modifications to a client processing system and at least one file required to implement at least one modifying action, said tool comprising:

a plurality of classes, each class corresponding to a respective type of action wherein the software package comprises a hierarchical structure of leaf and branch nodes capable of being traversed in a top-down manner and each leaf node corresponding to said respective type of action;

means for reading said software package file and instantiating a class having attributes corresponding to the respective type of each of the at least one action of said software package file and setting the attributes of the at least one class according to the respective action definition in said software package file;

means for executing a check method on at least one of each of said at least one class instances to determine if a deployment operation can be implemented in a specified first mode;

means, responsive to a successful check, for executing a first method on each of said at least one class instances in said first mode, said first method stored on the computer readable storage medium; and

means, responsive to check failure of any class instance, for executing a second method on each of said at least one class instances in a second mode, the second mode being less preferable than the first mode, said second method stored on the computer readable storage medium.

Claim 2. A software deployment tool according to claim 1 wherein said deployment operation is one of installation or removal of a software package.

Claim 3. A software deployment tool according to claim 1 wherein said first mode is an undoable mode and said second mode is a basic mode.

Claim 4. A software deployment tool according to claim 1 wherein said first mode is a basic mode and said second mode is a transactional mode.

Claim 5. A computer program product comprising computer program code stored on a computer readable storage medium for, when executed on a computing device, processing a software package file, the program code comprising the software deployment tool of claim 1.

Claim 6. A system for deploying software over a computer network, the system comprising:

a software package including at least one file, the file including definitions of actions involved in a software distribution, wherein the software package includes a hierarchical structure of leaf and branch nodes capable of being traversed from parent to child in a top-down manner;

a management agent configured to receive the software package; and

a target endpoint including a software package engine resident on the target endpoint, the software package engine configured to receive instructions via the management agent.

Claim 7. The system of claim 6, wherein the software package engine is further configured to decode the software package file.

Claim 8. The system of claim 6, further comprising a package editor configured to graphically define the software package.

Claim 9. The system of claim 6, wherein the hierarchical structure of the software package is serialized.

Claim 10. The system of claim 6, wherein the software package is represented by a sequence of stanzas, with each stanza representing an action.

Claim 11. The system of claim 10, wherein the stanzas are nested such that stanzas containing other stanzas represent a container action.

Claim 12. The system of claim 6, further comprising a preparation and test site configured to transform the software package from one format to another.